How to Protect Against a Militant Spammer

Markus Jakobsson* Joy Müller†

Abstract

We consider how to avoid unsolicited email – so called *spam* – in a stronger adversarial model than what has previously been considered. Our primary concern is the proposal of an architecture and of protocols preventing against successful spamming attacks launched by a strong attacker. This attacker is assumed to control the communication media and to be capable of corrupting large numbers of protocol participants. Secondly, we show how services such as message integrity and message privacy can be obtained virtually for free by using the structures put in place for spam protection. A final issue we are concerned with is how to produce protocols that have the property that the bulk of the computational requirements and storage requirements are shifted towards one of the participants, in order to reflect the desired power balance and to avoid denial-of-service attacks. This results in a simple and efficient solution that is largely backwards-compatible, and which addresses many of the concerns surrounding email communication.

Keywords: authentication, light-weight, spam, strong adversary

1 Introduction

To many people, email is becoming a crucial tool of daily life, much like the car became an integral part of life for many families some decades ago. Email, in fact, to some extent is a substitute for the car, in terms of allowing telecommuting and quick delivery of information. For both of these vehicles of the twentieth century, there is a need for traffic rules and enforcement of the same. Clearly, we would not tolerate a truck parked in the middle of an intersection, the driver handing out flyers. By the same token, there is no reason why the same behavior in the digital domain should be accepted. Until quite recently, though, there has been no suggestion of how to enforce good behavior on the digital roads.

*Information Sciences Research Center, Bell Laboratories, Murray Hill, NJ 07974, markusj@research.bell-labs.com

†Johannes Gutenberg Universität Mainz, Fachbereich Mathematik, 55128 Mainz, joy@dialup.nacamar.de

A recent upswing in the amount of *spam*, i.e., email broadcast advertisements, has caused both considerable congestion and concern, particularly in light of how inexpensive and efficient the spamming methods are, and the ease with which spammers have been able to avoid having their outgoing traffic filtered out, by moving from domain to domain and by disguising themselves. In [10], Gabber et al suggested a method for categorizing email, allowing unrequested email to be automatically erased without consuming any system resources or user time. However, their model assumes a rather "nice" adversary: The spammer in their model is only assumed to use today's methods of acquiring email addresses, namely to either buy lists of valid email addresses or collect these from newsgroups, etc. Given that their solution – if employed – makes the efforts of such a spammer completely meaningless, it is likely that the warfare is stepped up, and a new breed of more militant spammers becomes common.

It is the aim of this work to defend against this more aggressive type of behavior just as the spammer with today's methods was defended against in [10]. We propose a new, stronger adversarial model, and propose an efficient solution that meets the adversarial model and that makes the efforts of the militant spammer almost pointless. Here, the word "almost" is important: it is necessary to allow for normal email traffic, and allow two entities that have not previously established a communication link to communicate with each other (or the strength of email will be severely reduced), so it will still be possible for a spammer to send advertisements in a very well targeted manner. We mean that this is not unreasonable, though, since it shifts the effects of the spammer from an instigator of traffic jams to an advertiser that only sends very well directed advertisements in significantly lower quantities.

Before we discuss the new adversarial model, let us linger briefly on the model and solution of [10]. There, a spammer is assumed to obtain email addresses only by compromising servers holding valid addresses (this models both getting email addresses from newsgroups, and buying them from companies compiling email address lists, and similar approaches.) The solution suggested is to use *extended addresses*, which consists of the "normal" addresses (the so called *core* address) and an extension. The extension is a sequence of characters acting like a password for the email to be accepted, where the extension used by one pair or participants cannot be guessed from a set of other extensions. An extension may only be obtained by sending a request (of a special format that cannot be used for spam) to the desired receiver of an email message; the request may contain a proof of having performed a certain computational task, particular to the pair of sender and receiver (thus implementing a cost for each connection to be set up). Similarly, it may contain a proof of a monetary expense, such as an attached digital coin whose validity can easily be verified and which may be cashed as a "punishment" for spamming.

Consider now an attacker that is allowed to eavesdrop on communication lines, and who may collect valid extensions when he sees them, either in a response to a request for an extension, or more straightforwardly, from the

address fields of valid messages passing from the senders to the receivers. He may then use these extensions, possibly masquerading as the sender he "stole" the extension from. Even though it clearly makes the compilation of address lists more elaborate, the attack is far from difficult to perform; in fact, the only reason why this approach is not used today already is probably that it is slightly simpler to collect addresses from newsgroups, etc.

An even more aggressive attacker is allowed the following, ultimate, attack as well: he may, apart from eavesdropping on the communication lines, also perform substitution attacks, viz. substitute a legal email with his advertisement (wherein he may keep parts of the old message, such as the sender and receiver information, and the extension employed therein.) In other words, this militant breed of spammer may monitor network traffic, and adaptively insert and substitute messages, hoping to succeed in making a receiver accept a message that was not sent from a party from whom the receiver desires email.

We suggest an efficient and structurally very simple solution that protects against an adversary of the above type. It relies only on very conventional and well analysed cryptographic functions. Our solution is efficient in terms of storage, communication, and computation. Moreover, and also importantly, the solution is largely backwards-compatible with existing methods, and allows for a gradual implementation (in the sense that some entities may choose to employ the scheme, while others do not.)

Note that we consciously shift the computational and storage requirements towards the senders of the emails, in order to make the load of the receivers lighter. This serves a dual purpose: First, to discourage spam, and second, to limit the risk of a denial-of-service attack (see e.g., [14] for related issues and methods).

Organization: We present our goals in section 2, and review related work in section 3. After specifying the model and requirements in section 4, we list what building blocks will be used in section 5, and finally we present our solution in section 6. In the Appendix, we prove our solution to satisfy our requirements based on the assumptions.

2 Goals

We will now informally describe the desired requirements we want our solution to satisfy, given the adversarial model to be considered:

Only messages originating from *registered* senders will be delivered; all other messages will be weeded out at the gateway of the receiver. A party becomes registered by paying an agreed-upon price, which may be either a computational or monetary cost. It must not be possible to substantially reduce the price per working connection. (This allows us to establish lower bounds on the costs of spamming.) Furthermore, a message recipient can retract the registration of

any sender at any time, thereby forcing spammers to pay the price of becoming registered for each email message to be sent. The receiver may also implement a categorization of senders, allowing sorting of messages according to meaning, context and priority.

Note that our suggested solution will, as a side-effect, allow us to provide authenticity verification of the messages virtually for free. This means that the receiver will be able to determine whether the received message was indeed sent from the claimed sender. The authentication can be either relative to a pseudonym (for which the identity of the owner is not known), or relative to a certified identity. The former is trivially obtained using our proposed solution, and the latter can easily be obtained if the sender has a public key registered, allowing her to prove knowledge of the corresponding secret key during the setup phase (which only occurs once for each pair of participants.) Note also that the authentication feature does not allow the receiver to convince a third party of the authenticity of a given message (without the implicit cooperation of the sender), i.e., we obtain *private authentication*[1] of messages. (If this property is not desired, then standard digital signature methods can be applied on a per-message basis.) Whereas neither authentication nor private authentication are primary goals of our investigation, they constitute an additional bonus of employing our method. Finally, it is easy to obtain message privacy (using standard encryption methods) by employing the structures already used for the spamming protection. Therefore, our solution, while not cryptographically advanced, successfully manages to solve an array of related and functionally interdependent real-world problems, whose exact relations have not previously been clarified. The contribution of our work should be seen in terms of the practical issues it addresses, and in the clean and simple solution it provides.

3 Related Work

The conceptually perhaps simplest and least intrusive approach to spamming prevention, and the absolutely most commonly used method, is filtering according to senders' addresses and according to message and subject keywords. However, it is not very efficient, as it is easily foiled by adversaries avoiding particular keywords known to be black-listed, and avoiding to use (or camouflaging the use of) certain black-listed sender addresses or domains.

A solution that better prevents against spam is to use electronic mail channel identifiers [11]: A receiver assigns a different channel to each sender by giving each of them a unique email address at which to contact him. Incoming mail gets sorted or rejected according to the address sent to and the sender they originate from. Mails originating from unknown senders may be put in a "public" channel. (These public channels are therefore not spam-free.)

[1] See, e.g., [13] for a description and solution of a similar concept relating to signatures.

A related approach is to use secret extensions of email addresses (which in principle are sender-recipient unique passwords appended to the recipient user name) instead of different channels. This is done in the scheme by Gabber et al [10]. There, spamming-prevention is done by filtering the incoming mails according to the extension, discarding emails without a valid extension. The weaknesses of both [10] and [11] stem from the fact that the "passwords" are communicated in the clear, allowing an eavesdropping adversary to circumvent the security measures. A difference between [10] and [11] is that Gabber et al employ computational puzzles to inflict a computational cost on anybody who wants to be given a valid extension.

The idea to require computational payments was first proposed by Dwork and Naor [6]. This was also the first paper to consider how to prevent against spam using cryptographic methods. In [6], an email has to be accompanied by a correctly evaluated *medium-hard function* (later also called a *puzzle*) in order to be accepted. In contrast, the solution by [10] only requires this setup to be performed once for each pair of communicating participants, which allows the computational cost to be raised without causing communication havoc. (This is an advantage, as by requiring higher computational costs, spamming is further discouraged.)

After the introduction of puzzles in [6], different solutions have been studied by Franklin and Malkhi [9], Gabber et al [10], and Juels [14], each paper proposing variants with different properties. These papers also use the resulting functions in different contexts, ranging from advertising to prevention of denial-of-service attacks.

The spam problem was also given attention by Cranor and LaMacchia [5], who reviewed the previously mentioned solutions and presented an outline of some regulatory solutions from the legislative realm.

All of the above solutions allow an adversary to steal information from messages that allows him later to spam, or to replace portions of messages so that they *become* spam. We consider a stronger model than what previous work does, to avoid these problems.

A central building block in our scheme is the message authentication code (MAC for short). A MAC is a keyed one-way function of the input, where the secret key is known by both the generator and the verifier of the MAC, and the validity of the MAC (corresponding to the authenticity of its input) relies on knowing the secret key. We refer to [22] for a review of MACs. An early description of a MAC construction based on a hash function was made by Tsudik in [23]. Preneel and van Oorschot [19] pointed out some problems of this construction, which was later studied and strengthened by Bellare, Canetti and Krawczyk [2]. In the latter paper, a hash-function based MAC design is proposed and proven to be secure based on general cryptographic assumptions.

Another building block in our scheme is the one-time pad. We use this for a realization of encrypted return-channels that allows the computational burden to be shifted from the recipient of a message to the originator, and allows the

recipient to return encrypted messages with a minimum of effort. Note, however, that the one-time pads are generated as the outputs of a pseudo-random function and communicated encrypted using standard public key methods. (Thus, the one-time pad is only as secure as its building blocks, and is not information-theoretically secure as one-time pads are when truly random and communicated out-of-band.)

4 Model and Requirements

In our model, participants are modelled as senders and receivers, where each participant may alternatingly play both of these roles. We denote the sender Sally with S, and her intended receiver Ray with R. In our model, a participant is therefore a computational device, and this can be modelled by a polynomial-time Turing Machine. A *user* is the human or program with access to the machine that corresponds to a participant. For a user to be presented with an email M along with a label describing its sender S, M must first be *accepted* by the receiver R corresponding to this user. In turn, for a receiver R to accept an email M from a sender S, an entry describing the latter needs to be in a list of wanted senders with R. Descriptions of participants get added to this list by paying a setup cost C, but may be removed from the list by R if R (or its corresponding user) so decides. When this occurs, no more messages from S will be accepted by R, and no more messages from S will be displayed to the user (including those already accepted but not already displayed.)

Adversary. An adversary is a party who may corrupt any number of participants. When a participant is corrupted, the entire memory contents and computational resources are made available to the adversary, who also controls what messages are sent by the corrupted party. It is assumed that the adversary can not influence the actions of a participant who is not corrupted. Additionally, the adversary controls the communication channel, and may remove and inject messages at will. He can read all communication between non-corrupted participants. We model the latter by the (stronger) adversarial model in which the adversary may chose a polynomial number of messages and have the MACs of these computed by an oracle before he attempts to perform an attack. (Note, however, that the adversary may not include one of the MACs obtained from the oracle in any of the messages he causes to be sent.) The adversary has the following goals:

Spamming attack: One goal of the adversary is to make some k non-corrupted participants $\mathrm{R}_1 \ldots \mathrm{R}_k$ accept each one message $M_1 \ldots M_k$, where these messages are chosen by the adversary, without the adversary having to pay a total cost close to or exceeding $(k - k')C$, where k' is the number of the above messages that are labelled as being sent by a participant who is corrupted.

Authentication attack: A second goal of the adversary is to make a receiver R accept an email M and present this to the user as originating from a non-corrupted participant S, when in fact the message did not originate from S.

Sender privacy attack: A third goal of the adversary is to convince a non-corrupted participant that a message M was sent by a non-corrupted sender S to some (potentially corrupted) receiver R.

Message privacy attack: The fourth and final goal of the attacker is to determine whether a first message M_1 or a second message M_2 was sent between two non-corrupted participants. More formally, this corresponds to a successful chosen ciphertext attack (see, e.g., [3] for a detailed treatment.)

We require that an adversary cannot succeed with the above attacks with more than a negligible probability. More specifically, except with a negligible probability, an adversary cannot (1) succeed with a *spamming attack*; (2) succeed with an *authentication attack*; (3) succeed with a *sender privacy attack*; or (4) succeed with a *message privacy attack*.

5 Building Blocks and Assumptions

Authentication: As an authentication mechanism we use so-called message authentication codes, which we use both to *validate* and *authenticate* the transmitted messages. The MAC can be based on a keyed cryptographic hash function h, where the key is known (only) by the sender and the intended receiver(s). We require that it must be a hard problem to determine whether a given string is a valid MAC on a given message m, unless the verifier has access to the secret key. This must hold even after a polynomial number of MACs for chosen messages (other than m) have been seen. The security of the authentication mechanism relies on the assumptions on the underlying hash function. It has been shown that for this to hold, it is sufficient and necessary that it is hard to compute the output of the hash function without knowledge of the secret key, and that the hash function must be (weakly) resistant to collision finding. (We refer to [1, 2] for a thorough treatment.)

In addition to these basic properties, we assume that the hash function h can be modelled by a random oracle [4], in order to use it for key generation purposes. (This assumption may be replaced by another if we use another method for key generation.)

In practice, it is believed that MD5 [20] and SHA-1 [17] have the desired properties required for the hash function.

One-Time Pad: P denotes a one-time pad, which is generated by a pseudo-random function generator, e.g., using a keyed hash function h on an input which is a unique serial number.

Symmetric Encryption: To achieve privacy we use a symmetric encryption, such as the DES [15] or the proposed AES. We assume the symmetric encryption to be secure against chosen cipher text attacks.

Public-Key Encryption: For the protocol we use a public-key encryption scheme that is secure against adaptive chosen message attacks [18]. Such a scheme could be RSA [21] or ElGamal encryption [8].

Setup cookies: For each setup, the initiator sends over a *setup cookie* that is evidence of the initiator having paid a cost (or that it is possible to force him to pay this cost.) This may either be implemented using a puzzle, or by some monetary mechanism.

6 Design Principles and Solution

A first design principle is to shift the computational burden from the receiver of an email to the instigator. In order to do this, we want the setup cookie to be verifiable with a minimum of effort by the receiver. In particular, we do not want the receiver to have to perform any decryption for this to be possible, and so, since the setup cookie must be sent in cleartext, it must be specific to the sender and receiver (in order to avoid "cookie kidnappings".) Furthermore, the response should also be possible to generate with a minimum of effort, which is the reason for the *one-time response pad* we suggest. Finally, we want to minimize the amount of storage allocated by the recipient, which is done using the compacting method suggested in [12]. All together, the use of these principles reduce the risks for denial-of-service attacks mounted on potential receivers by flooding these with incorrect setup requests.

A second design principle is for valid extensions to depend on the message and be such that they cannot be forged by an eavesdropper. This corresponds to not sending the password in the clear, but to instead prove knowledge of the password, where the proof is done with respect to the message sent. This is practically achieved using MACs.

We now present our solution:

1. **Request for Setup:**
   If Sally sends an email to Ray's core address, her computer gets an automatic response with Ray's public key y_R in. Her email is not delivered, but bounces back. Along with the returning message, a homepage or ftp address may be sent, from which appropriate software may automatically be downloaded.

2. **Setup:**
   Sally's computer generates and sends a setup cookie and a one-time pad P to Ray's computer, the latter encrypted with Ray's public key y_R. Ray's computer verifies the correctness of the cookie (corresponding to verifying

that the puzzle solution or the piece of digital cash is valid), and selects a symmetric key K_{RS} uniformly at random from the set of possible keys. (We explain later how this can be performed in a way that conserves resources for Ray.) Ray adds a sufficient amount of redundancy[2] to K_{RS}, resulting in the value K'_{RS}. Ray then replies to Sally with $P \oplus K'_{RS}$ (using a publicly known extension used solely for setup). On receiving this message, Sally computes K_{RS} and verifies that no errors were introduced. She then stores (Ray, K_{RS}) in a list of all such access keys. All future emails from Sally to Ray will be processed using this key[3].

3. **Sending a message:**
   Let m be the message Sally wants to send to Ray, $\mu = hash(m)$, and let $e = MAC_{K_{RS}}(\mu)$. This value e is used as an extension of Ray's email address when the message m is sent to him.

4. **Receiving a message:**
   Ray's computer looks up (or computes) K_{RS}, given the alleged sender of the email, Sally. Ray's computer calculates e as above, and accepts the email iff the same result as in the extension is obtained. An accepted email gets delivered to Ray, just like a normal email would normally be delivered after having been received. Otherwise, the email is considered as a request for setup (see above) and the message goes undelivered.

Remark 1: Compact key management. The symmetric key can be generated by Ray as $K_{RS} = h(K_R, S, i_S)$, where h is a hash function, K_R is Ray's secret key, S is Sally's name and i_S is a sequence number that is increased for each setup request by Sally. In our model, this amounts to the same as choosing this key uniformly at random from the same set of keys. However, it allows Ray a very compact representation of the key; he merely has to store his secret key and a small counter indicating the value of i_S.

Remark 2: Message Privacy. For the message encryption we propose to use DES in the CBC mode [16]. The secret key can be generated by Sally as $K_S = h(K_{RS}, count)$[4], where h is a hash function, K_{RS} is the shared key from the MAC, and *count* is a counter increased for each message sent between

[2]The redundancy is added in order to allow the detection of third-party mounted substitution attacks. This is an attack which, if successful, manages to convince Ray that he has received a valid key from Sally, whereas he has not. This in turn will make his future messages to Sally either bounce or be discarded (depending on how the system is configured.) The redundancy adding function must be such that the resulting encryption scheme is non-malleable [7]. We suggest adding redundancy using an error detecting code. Due to space limitations, we do not, however, elaborate on how to do it.

[3]We make the simplifying assumption here that the scheme is symmetric, i.e., all future emails from Ray to Sally will use this key as well.

[4]It is also possible to use the key K_{RS}, but to achieve stronger security and easier analysis, we propose to use different symmetric keys for the two applications.

Sally and Ray. (We note that this key may be generated without any extra communication.)

Remark 3: Categorization. In the above, we have only described filtering to avoid unwanted email. It is trivial to use the same methods to categorize the incoming emails according to sender-specific categories. It is possible for a sender to request a higher (or lower) priority by indicating this in some field that gets authenticated along with the message. This is relatively safe from abuse since if a user is considered to have abused the increased priority setting he may lose his registration.

Remark 4: Use of Headers. Instead of using the format of address extensions suggested in [10], the MAC can be communicated in an available header of the email. This allows an increased transparency of the implementation.

Remark 5: An alternative implementation. Instead of using the above method in which the reply is encrypted using the previously transmitted one-time pad, we can have Sally send over a public key which is used by Ray to encrypt the symmetric key sent to Sally. In order to minimize the computation for Ray, RSA should be used for this purpose. In order to avoid the replacement of K_{RS} by a saboteur adversary, Ray could sign the key before encrypting it. All in all, this solution appears to amount to roughly the same computational expenses for Ray.

This basic protocol satisfies requirements 1-3, and requirement 4 as well for the protocol in which encryption is used to obtain message privacy. We prove these claims in the Appendix.

Acknowledgements

Many thanks to Daniel Bleichenbacher for inspiring discussions.

References

[1] M. Bellare, R. Canetti, H. Krawczyk, "Message authentication using Hash Functions-The HMAC Construction," RSA Laboratories' CryptoBytes, 1996.

[2] M. Bellare, R. Canetti, H. Krawczyk, "Keying Hash Functions for Message Authentication," Advances in Cryptology - Proceedings of Crypto'96, pp. 1-15.

[3] M. Bellare, A. Desai, E. Jokipii, P. Rogaway, "A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation," FOCS, 1997.

[4] M. Bellare, P. Rogaway, "Random oracles are practical: A paradigm for designing protocols," First Annual Conference on Computer and Communications Security, ACM, 1993.

[5] L.F. Cranor, B.A. La Macchia, "Spam!," Communications of ACM 98. Available at `http://www.research.att.com/~lorrie/pubs/spam`.

[6] C. Dwork, M. Naor, "Pricing via Processing or Combating Junk Mail," Advances in Cryptology-Proceedings of Crypto'92, pp.139-147.

[7] D. Dolev, C. Dwork, M. Naor, "Non-malleable cryptography," STOC'92, pp. 542-552.

[8] T. ElGamal, "A Public-Key Crytosystem and a Signature Scheme Based on the Discrete Logarithm," Advances in Cryptology - Proceedings of Crypto'84, pp. 10-18.

[9] M. Franklin , D. Malkhi, "Auditable Metering with Lightweight Security," Financial Cryptography '97, pp. 151-160.

[10] E. Gabber, M. Jakobsson, Y. Matias, A. Mayer, "Curbing Junk E-Mail via Secure Classification," Financial Cryptography '98.

[11] R.J. Hall, "Channels: Avoiding Unwanted Electronical Mail," Communications of ACM '98. Available at `ftp://ftp.research.att.com/dist/hall/papers/agents/channels-long.ps`

[12] M. Jakobsson, "Mini-Cash: A Minimalistic approach to E-Commerce," PKC '99, pp. 122-135.

[13] M. Jakobsson, K. Sako, R. Impagliazzo, "Designated Verifier Proofs and Their Applications," Advances in Cryptology-Proceedings of Eurocrypt'96, pp. 143-154.

[14] A. Juels, J. Brainard, "Client Puzzles : A Cryptographic Countermeasure Against Connection Depletion Attacks," Proceedings of the 1999 Network and Distributed System Security Symposium, Internet Society, pp.151-165.

[15] NBS FIPS Pub 46-1, "Data Encryption Standard," U.S. Department of Commerce, 1988.

[16] NBS FIPS Pub 81, "DES modes of operation," U.S. Department of Commerce, 1980.

[17] NBS FIPS Pub 180-1, "Secure Hash Standard," U.S. Department of Commerce, 1995.

[18] M.Naor, M. Yung, "Public-key cryptosystems provably secure against chosen chipertext attacks," Proceedings of the 22nd Annual Symposium on Theory of Computing, ACM, 1990.

[19] B. Preneel, P. van Oorschot, "On the security of two MAC algorithms," Advances in Cryptology-Proceedings of Eurocrypt'96, pp. 19-32.

[20] R. Rivest, "The MD5 Message-Digest Algorithm," RFC1321, 1992.

[21] R.Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, v.21, n.2, 1978, pp. 120-126.

[22] B. Schneier, *Applied Cryptography.*

[23] G. Tsudik, "Message authentication with one-way hash functions," ACM Computer Communications Review,v.22,n.5,1992, pp. 29-38.

A Proofs

Theorem 1: Our system protects against the *spamming attack* in the chosen message attack model. This means that we first allow an adversary E to receive valid MACs on a polynomial number of messages that E chooses. Then, if an adversary is successful in making some k non-corrupted participants $R_1 \ldots R_k$ accept each one message $M_1 \ldots M_k$, he must first pay at least a cost $(k-k')(C-\epsilon)$, where k' is the number of the above messages that are labelled as being sent by a corrupted participant, and ϵ is a small constant that corresponds to the maximum savings possible by batching several setup computations.

Proof of Theorem 1:
Let us assume that the receiver R accepts an email with the belief that it was sent by a sender S. Since the receiver is assumed not to be corrupted, he only accepts emails with valid MACs w.r.t. the believed sender. The adversary E cannot gain any information from corrupting participants not involved as sender or receiver in any of the message transfers, since the secret keys used for the MACs are chosen independently at random. (If this does not hold then the assumption that the hash function used to generate the shared secret keys is a random oracle does not hold.) Moreover E can not get any information about the secret key during its transmission, since the one-time pad is secure because its encryption protects against adaptive chosen message attacks. Therefore, the adversary must either (1) corrupt somebody who has paid the setup cost C, or (2) pay the setup cost $C-\epsilon$ for the email to be accepted, or (3) has to produce the valid MAC without the key given in the setup phase. (This is the case since no such key or portion of it will be given out without the expected setup cost C has been paid.) In the former case, S is one of the k' corrupted players, and in the second S is one of the k "paying" participants. We argue that the third case only can occur with a negligible probability. This is so since the MAC according to the assumptions is resistant against chosen message attacks. Thus, the total cost paid by the adversary is $(k-k')(C-\epsilon)$. □

Theorem 2: Our system protects against the *authentication attack*, i.e., the adversary is not able to make a receiver R accept an email M and present this to the user as originating from a non-corrupted participant S, when in fact the message did not originate from S.

This follows from the proof of Theorem 1, and the fact that non-corrupted participants will only accept emails with valid MACs corresponding to the apparent sender.

Theorem 3: Our system protects against the *sender privacy attack*, i.e., an adversary is not able to convince a non-corrupted participant E that a message M was sent by a non-corrupted sender S to some (potentially corrupted) receiver R.

Proof of Theorem 3: Due to the property that the MACs are assumed secure against a chosen message attack, it is not possible for a third party to determine that a given message was sent by one of R and S (without attempting to determine which one) without the involvement by either of these two parties. Let us therefore assume that one of these two parties collaborates with the third party to try to convince the latter that the other participant sent a certain message. If M was sent from S to R than it has a valid extension computed with the correct key K_{RS}. However, this key is also known to R, and thus, it is not possible to determine whether R or S originated the message. Therefore, if one of the participants were to try to convince a third party of who the sender is, the third party will not trust the proof, as the transcript could have been computed by either R or S. □

Theorem 4: If the encryption mode of our scheme is employed, then our system protects against the *message privacy attack*, i.e., the attacker is not able to determine whether a first message M_1 or a second message M_2 was sent between two non-corrupted participants.

This follows directly from the symmetric cipher being assumed to be secure against chosen message attacks.

SUBMISSION NO. : 118626

ATTORNEY : Weinick, Jeffrey M.

Title :
How To Protect Against A
Militant Spammer

----------MAIN INFORMATION----------

ITEM STATUS	: Opened	LUCENT RATING	:
STATUS DATE	: 04/23/1999	PATENT WPN	: U0
OPEN DATE	: 04/05/1999	GOVT. CONTRACT	: No
CLOSE DATE	:	TYPE	: Patentability
DEADLINE DATE	:	DEFENSIVE	: No
BU CODE(S)	: BLRS,OUTSIDE		

----------SUBMITTER INFORMATION----------

SUBMITTER NAME	: Jakobsson, Bjorn Markus
COMPANY	: LUCENT
LOC_EXT	: 908-582-8138
DEPARTMENT	: BL0112350
DIRECTOR	: A. Silberschatz

SUBMITTER NAME	: Mueller, Joy Colette
COMPANY	: UMAINZ
LOC_EXT	:
DEPARTMENT	: OUTSIDE
DIRECTOR	:

Brief Description :

No data in this field